ELSEVIER Nuclear Instruments and Methods in Physics Research A 404 (1998) 445–454

NUCLEAR INSTRUMENTS & METHODS IN PHYSICS RESEARCH Section A

Pattern recognition of particle tracks using principal component analysis and artificial neural network

D. Dutta[a], A.K. Mohanty[a,*], R.K. Choudhury[a], Phool Chand[b]

[a] *Nuclear Physics Division, Bhabha Atomic Research Centre, Mumbai 400085, India*
[b] *Computer Division, Bhabha Atomic Research Centre, Mumbai 400085, India*

Received 29 May 1997; received in revised form 26 September 1997

Abstract

A new method is suggested for pattern recognition of particle tracks based on a combined approach of both artificial neural network (ANN) and principal component analysis (PCA). It is seen that in high multiplicity environment, neither the PCA nor the ANN method is satisfactory when used separately as a track classifier. Best performance is achieved when the data are preprocessed using PCA technique, before it is fed to the backpropagated neural network. © 1998 Elsevier Science B.V. All rights reserved.

1. Introduction

Pattern recognition and track reconstruction is one of the important aspects in the detectors used in high energy heavy ion experiments. When the track multiplicities are high and the detectors are placed in the inhomogeneous magnetic fields, the track finding algorithm becomes very complex. In many situations, the tracks might also suffer multiple scattering. It is, therefore, essential to develop some efficient algorithm taking into account the detector geometry, the inhomogeneity of the magnetic field surrounding the detectors, multiple scattering and so on. In this work, we have tried to find out a track finding technique that will be well suited in tracking environments where one expects large track multiplicities, particularly in case of relativistic heavy ion collisions. We apply the methods to the muon tracking system of the PHENIX detector, which is being designed [1,2] for experiments with the Relativistic Heavy Ion Collider at BNL. The main purpose of the PHENIX muon arm detector is to detect muon pairs with high mass resolution originating from the vector meson decays or virtual photon production. Therefore, a highly efficient pattern recognition technique is desirable as the momentum or the mass resolution of the muon pairs will depend on how well the tracks are identified from the set of measured co-ordinates. We generate simulated data for PHENIX muon arm using GEANT based simulation code PISA [3]. Conventionally, the Principal Component Analysis (PCA) method is used for pattern recognition and classification of true tracks [1,4]. It is found that the PCA method works well when the tracking distance is small, number of tracking stations are large and the track

* Corresponding author.

multiplicity is not too high. In this work, we show that the PCA method is not good enough in the high multiplicity environment, since many combinations of the co-ordinates not belonging to the true tracks also qualify the PCA test. Next, we consider another approach based on Artificial Neural Network (ANN) which is recently becoming quite popular in many of the classification applications [5]. We found that the ANN method based on the normal backpropagation algorithm does not work well in the present application and it even does not converge while learning. To overcome the problem of both PCA and ANN, we use a simple neural network with preprocessed inputs. It is shown that best performance is achieved when the input data is preprocessed using the PCA technique before it is fed to the backpropagated neural network. The advantage of this network is that it does not have much connections in the hidden layer and thus the complexity of error propagation through the hidden layer is avoided. In the following, we work with the simulation data generated for the muon arm of the PHENIX detector, to make a comparative study of the above track finding algorithms. We have carried out GEANT based simulation for tracking of the muons detected in the PHENIX muon arm. The aim of the present study is to demonstrate how a simple neural network can be used with preprocessed inputs obtained from PCA as a classifier, particularly in a situation where the track multiplicity is quite high.

2. Simulated test data

Fig. 1 shows a schematic drawing of the PHENIX muon north arm. Particles emerging from the collision point or vertex point pass through the pole tip of the central magnet (CM) and enter the muon arm. The muon arm momentum measurement uses the radial magnetic field produced by the muon magnets (MM) having a central piston and flux return plates which resemble a "lamp shade" geometry. Charged particles entering the muon arm are tracked by multi-layer tracking chambers arranged in three stations located at distances of 1.8, 3 and 6 m from the vertex position. Downstream of the MM, muons are distinguished from pions and other shower products by means of muon identifier detectors,

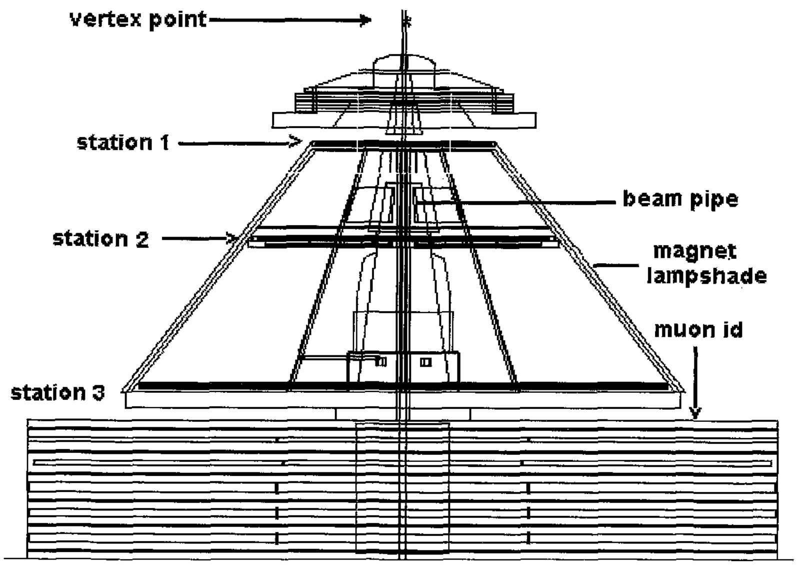


Fig. 1. Schematic diagram of PHENIX muon north arm.

consisting of concrete absorber walls interleaved with planes of limited streamer tubes. The first tracking station is out side the magnetic field whereas the middle and the last one lie inside the magnet lamp shade. The muon arm is designed to give an acceptance for particles decaying into lepton pairs emitted in the rapidity range of $y = 1.2$–2.4. In the high energy limit, the rapidity y is related to the polar angle θ by $y = -\ln(\tan \theta/2)$. The above rapidity range corresponds to the polar angles of $10° \leqslant \theta \leqslant 35°$. The muon detectors have nearly full azimuthal angle coverage. The muon tracking detectors are made of cathode strip chambers having three planes in cathode–anode–cathode configuration. Each tracking station consists of three layers of cathode strip chambers, each of which provides information on the x and y co-ordinates of the particle track. More details on the design of the detectors are given in the Conceptual Design Report of the PHENIX detector [1]. The PHENIX detector has been modelled using GEANT based packages. We generate the test data for PHENIX muon tracking stations using PISA (PHENIX INTEGRATED SOFTWARE ANALYSIS) simulation software with realistic active and passive detector volumes and also with magnetic field on. PISA is a GEANT based simulation code [3] which gives raw detector hits; in this case the x, y and z co-ordinates at each layer of the tracking station. We used an in built J/ψ generator that produces J/ψ's which decay into a pair of muons. Fig. 2a shows a typical spectrum of the momentum distribution of the muons obtained from PISA simulation at layer one, i.e. just before station 1 for all those muons which enter the muon arms and penetrate all through up to the station 3, while Fig. 2b shows the rapidity distribution of the muons. In the first choice, we take only those muon pairs which penetrate all the three tracking stations. Although, we have chosen muon pairs to start with, the following analysis will also be applicable to pion tracks that penetrate all the three tracking stations. To see how the method works, we have considered this simple case which will be generalized to more complex situations later on. While passing through the three layers of a tracking station, a single track will create nine space points and there will be a total 27 of space co-ordinates while passing through all three stations. Since z positions are fixed, we are left with 18 (θ, ϕ) or (x, y) co-ordinates.

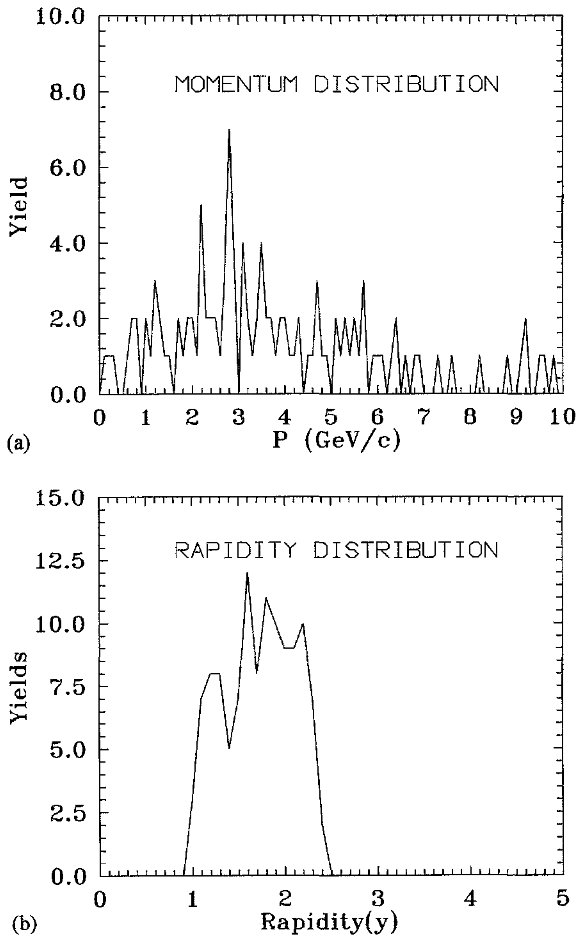


Fig. 2. (a) Momentum distribution of muons before station 1. (b) Rapidity distribution of muons that enter the PHENIX muon arm.

3. Principal component analysis

The principal component analysis which is also called a "canonical transformation" method does a simple co-ordinate transformation to principal axes such that the variances of the new co-ordinates are equal to the eigenvalues of the co-variance matrix, which is generated out of a large sample space [4]. Let us assume that a single track is characterized by a M-dimensional hit vector $\boldsymbol{h} = (h_1, h_2, \dots, h_M)$ whose average (H_i) and covariance (A_{ij}) are

$$H_i = \frac{1}{N} \sum_{n=1}^{N} (h_i)_n, \tag{1}$$

$$A_{ij} = \frac{1}{N} \sum_{n=1}^{N} [(h_i)_n - H_i][(h_j)_n - H_j], \tag{2}$$

where N is the number of hit vectors of the $(N \times M)$ dimensional sample space. Consider the following linear transformation:

$$X_j = \sum_{i=1}^{M} \omega_{ij} h_i. \tag{3}$$

It can be proved [4] that the variances of $\boldsymbol{X}$ are the eigenvalues λ of the dispersion matrix $\boldsymbol{A}$, when ω_{ij} represents the eigenvectors of A. Hence,

$$\mathrm{Var}(X_j) = E\left[X_j - E\left(\sum_{i=1}^{M} \omega_{ij} h_i\right)\right]^2$$
$$= E\left[\sum_{i=1}^{M} \omega_{ij}(h_i - H_i)\right]^2 = E(\eta_j^2) = \lambda_j, \tag{4}$$

where E stands for average over N and η_j is given by

$$\eta_j = \sum_{i=1}^{M} \omega_{ij}(h_i - H_i). \tag{5}$$

Therefore, the PCA is tuned with a training data set in order to calculate the average and the dispersion matrices $\boldsymbol{H}$ and $\boldsymbol{A}$, respectively. The experiment records a set of (θ, ϕ) pair from each layer when a charged particle enters the tracking station. Since track multiplicities are high, it is not possible to know which are the pair of co-ordinates from each layer that constitute a true track. Therefore, we consider all possible combinations $\boldsymbol{C} = (C_1, C_2, \dots, C_M)$ where $C_1, \dots, C_M$ are the co-ordinates drawn randomly (but pairwise) from each layer. More details of how the combination vector is formed is given in Section 4. To examine if the combination $\boldsymbol{C}$ constitutes a true track, the following generalized distance is calculated:

$$\eta_j = \sum_{i=1}^{M} \omega_{ij}(C_i - H_i), \tag{6}$$

$$d = \frac{1}{M} \sum_{j=1}^{M} \frac{\eta_j^2}{\lambda_j}. \tag{7}$$

It is now required that for a track candidate, d is less than some maximum value $d_{\max}$. From the previous discussions we know that $E(\eta_j^2) = \lambda_j$ and $E(\eta_j) = 0$. Hence, $\eta_j/\sqrt{\lambda_j}$ will be distributed normally with unit standard deviation. In other words, if we chose $\eta_j/\sqrt{\lambda_j} < 3$, i.e. $d_{\max} < 9$, it means three standard deviations away from the mean which will occur only in 99.85% of the cases.

4. PCA results

As mentioned before, the PHENIX muon arm consists of three tracking stations, each having three layers. Therefore, a single track which reaches up to the third station will have nine pairs of θ and ϕ co-ordinates, i.e., in a total of 18 space co-ordinates. Since the measurements of the co-ordinates are independently done at each station, one needs to take all possible combinations of these co-ordinates and calculate the generalized distance "d" for them. If the track multiplicity at station 1 is N, this results in N^9 combinations. This needs enormous computing time, particularly when N is quite large. An alternate approach is to find out the track segments within each station and finally to make a global search to join all the track segments. This procedure will result in $3 \times N^3$ combinations. The finding of track segment proceeds according to the following scheme. First, we search for the tracks in a single tracking station ($M = 6$) and finally we make a global search ($M = 18$) to join the track segments. We follow this approach to reduce the computing time. We use a training data set with 116 hit vectors h, generated from PISA simulation. The test data set with different random combinations $\boldsymbol{C}$ is generated for different multiplicities N out of this training data set. Using the training set, the ω_{ij} and λ_j were calculated for each station as well as for the global search. Fig. 3 shows a typical

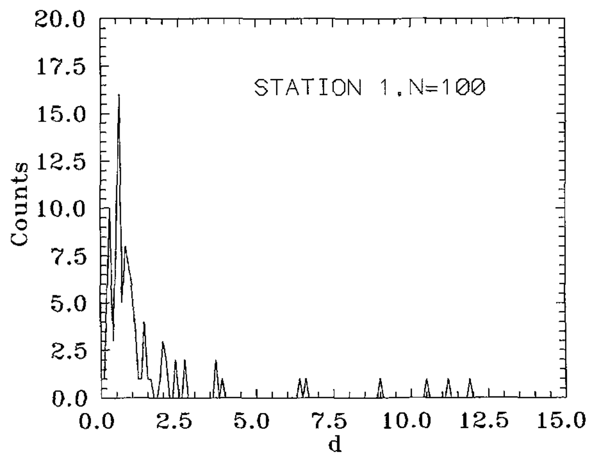


Fig. 3. The normalized distance "d" distribution for tracking station 1.

example of the "d" distribution for station 1 when same data set is used for both training and testing for a track multiplicity of $N = 100$. If one uses $d_{\max}$ around 16 (within 4σ) for the test data, all the true tracks will be accepted with 100% efficiency. Similar behavior is also found for stations 2 and 3. It is important to note here that if there is any combination of the vector C which is not a true track, but its "d" value is less then $d_{\max}$, it will also add to the true track counting. However, this does not happen within a tracking station. Table 1 summarizes the results for track recovery at all the three stations at various $d_{\max}$ values. The first column shows $d_{\max}$ values, whereas the second column shows the track multiplicities at each station. The third, fourth and fifth columns show the total number of tracks recovered at each station, i.e., the total number of tracks at each station for which $d \leqslant d_{\max}$. It is interesting to see that within each station, all the tracks for which $d \leqslant d_{\max}$ are true tracks and there is no false contribution as shown in the last column. As seen from the table, for $d_{\max} \approx 12$, the true track recovery efficiency reaches up to 100% for all stations. It is also seen that when track multiplicity is less, even at a $d_{\max}$ of 4 or 9 (within 2σ or 3σ) is good enough for nearly 100% recovery.

Next, we join these track segments doing a global search. Fig. 4 shows the "d" distribution of the true tracks for global tracking. In this case the value of d can reach as large as 60. In order to have 100% track construction efficiency, one must use $d_{\max} \leqslant 60$. However, in case of global tracking, an additional problem arises due to presence of many false tracks. There are many false combinations for which $d \leqslant d_{\max}$ and this happens at all $d_{\max}$ values,

Table 1
PCA performance in station nos. 1, 2 and 3

$d_{\max}$	Multiplicity	Station no. 1 Tracks recovered	Station no. 2 Tracks recovered	Station no. 3 Tracks recovered	False tracks
3.0	100	91	96	97	0
3.0	75	68	72	73	0
3.0	50	46	48	49	0
3.0	25	23	24	24	0
4.0	100	94	96	98	0
4.0	75	70	72	74	0
4.0	50	48	48	49	0
4.0	25	24	24	24	0
5.0	100	94	97	100	0
5.0	75	70	73	75	0
5.0	50	48	49	50	0
5.0	25	24	24	25	0
12.0	100	100	100	100	0
12.0	75	75	75	75	0
12.0	50	50	50	50	0
12.0	25	25	25	25	0

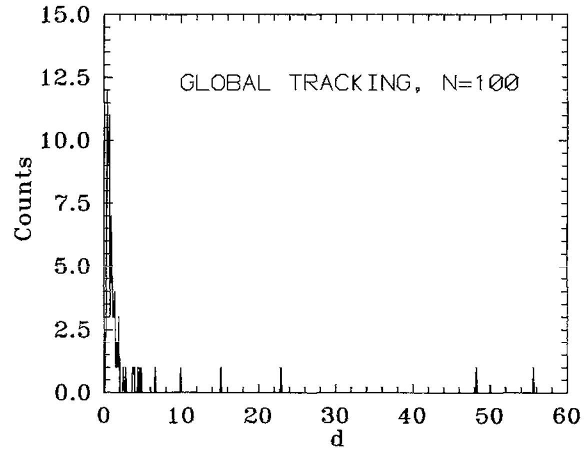


Fig. 4. The normalized distance "d" distribution for global tracking.

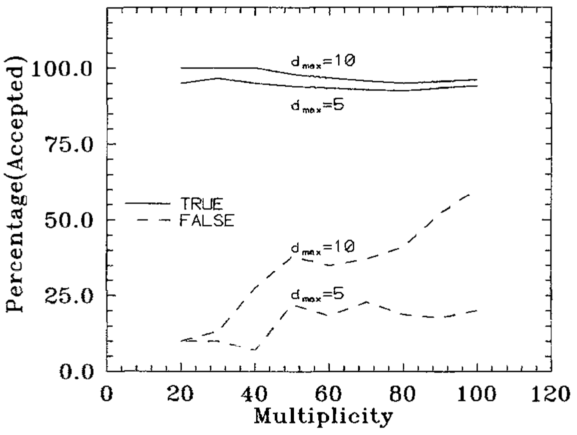


Fig. 5. The percentage of accepted tracks versus multiplicity for two different d_{max} values obtained from PCA method.

although the problem is more serious at higher d_{max} values as well as for higher track multiplicities. Fig. 5 shows the percentage of accepted true and false tracks as a function of track multiplicity at two different d_{max} values. Table 2 gives the results on the number of true and false tracks at multiplicity $N = 100$ for different d_{max} values. This shows that for $N = 100$, if one wants 100% true track efficiency, one needs to use a $d_{max} \approx 50$, which will also result in a large number of false tracks. Although this problem is less serious at low multiplicities, still it will result in lots of spurious tracks depending on the d_{max} value.

The contribution from false tracks might get reduced if the number of tracking stations is increased further. To see how the number of layers per tracking station affects the pattern recognition efficiency, we have removed one layer from each station. This configuration will result in 12 (θ, ϕ) co-ordinates for a track that goes all through. The results are also shown in Table 2. It is interesting to note that whether it is 18 or 12 co-ordinates, the performance does not differ significantly (see Table 2 for 18 and 12 co-ordinates). In the present case, we have considered only the muon tracks. It will be also of interest to see what will happen if we

Table 2
Global search for three tracking stations with 18 co-ordinates,12 co-ordinates and 6 co-ordinates, Multiplicity $N = 100$.

d_{max}	18 co-ordinates			12 co-ordinates			6 co-ordinates		
	True	False	Total	True	False	Total	True	False	Total
2.0	86	4	90	87	6	93	93	1049	1142
3.0	88	12	100	91	15	106	—	—	—
4.0	91	19	110	92	20	112	97	2441	2538
6.0	94	21	115	95	28	123	99	3926	4025
8.0	95	39	134	—	—	—	100	5381	5481
10.0	96	60	156	95	64	159			
15.0	96	100	196	96	100	196			
20.0	97	133	230	97	137	234			
30.0	98	228	326	99	227	326			
50.0	99	393	492	100	395	495			

do tracking with pions as it may give additional tracks in between due to decays. The last three columns of Table 2 show the performance with only one layer per tracking station which results in total six co-ordinates for a single track. In this case the tracking efficiency is very poor as the number of false tracks increases drastically.

5. Artificial neural NET (ANN)

The neural network takes a different approach. Consider a three layer network as shown in Fig. 6, where the output units are denoted by O_i, hidden units by V_j and input units by ξ_k. There are connections ω_{ij} from the inputs to the hidden units and W_{ij} from the hidden units to the output units. Different patterns are labelled by the superscript μ, so input k is set to ξ_k^μ when pattern μ is being presented. Given pattern μ, hidden unit j receives a net input

$$h_j^\mu = \sum_k \omega_{jk}\xi_k^\mu \tag{8}$$

and produces an output

$$V_j^\mu = g(h_j^\mu) = g\left(\sum_k \omega_{jk}\xi_k^\mu\right). \tag{9}$$

Output unit i thus receives

$$h_i^\mu = \sum_j W_{ij}V_j^\mu = \sum_j W_{ij} g\left(\sum_k \omega_{jk}\xi_k^\mu\right) \tag{10}$$

and produces for the final output

$$O_i^\mu = g(h_i^\mu) = g\left(\sum_j W_{ij}V_j^\mu\right) = g\left(\sum_j W_{ij} g\left(\sum_k \omega_{jk}\xi_k^\mu\right)\right). \tag{11}$$

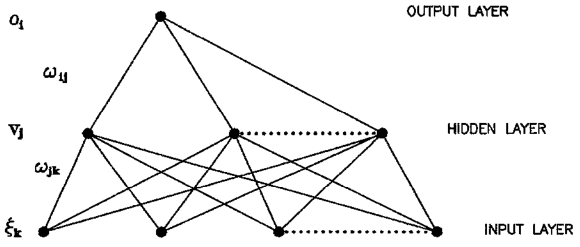


Fig. 6. A schematic diagram of a three layered neural network.

The net output O_i^μ for the pattern μ usually differs from the desired output d_i^μ. In the standard ANN model with back propagation algorithm [5], one tries to minimize the mean square error defined as

$$E = \frac{1}{2}\sum_{\mu i} [d_i^\mu - O_i^\mu]^2. \tag{12}$$

The aim is to adjust the free parameters of the network ω_{ij} and W_{ij} in such a way that the error function is minimum. Several methods have been proposed to speed up the training of the back propagation algorithm [6,7]. In this work, we find the ANN based on RPROP algorithm [6] is much faster than the one based on simple back-propagation.

6. Application of ANN for pattern recognition

We consider an ANN with 18 input nodes, 18 hidden nodes and one output node. The input and hidden nodes are kept same as this combination is found to give better performance. The output is assigned a value of either 0 or 1 depending on whether the input vector is a true or a false track. For unbiased training of the ANN, we take nearly equal number of true and false tracks (113 + 133) generated by the PCA method using certain $d_{\max}$ value. We select the input vectors in two different ways. In one case, we take θ and ϕ co-ordinates as input vectors, whereas in the second case we use PCA transformed co-ordinate d_j as input vectors where d_j is given by

$$d_j = \frac{1}{18}\frac{\eta_j^2}{\lambda_j} \tag{13}$$

such that $\sum d_j = d$.

Therefore, in the first case, the input vector has 18 actual θ, ϕ co-ordinates whereas in the second case, the input vector has 18 transformed d_j co-ordinates. The ANN is trained using JETNET-3.4 [8] in batch mode, i.e., the connection weights are updated after each epoch of 246 iterations. Fig. 7 shows the plot of error function after each epoch of 246 iterations with both (θ, ϕ) and d_j co-ordinates. It is seen that the learning is very poor with the

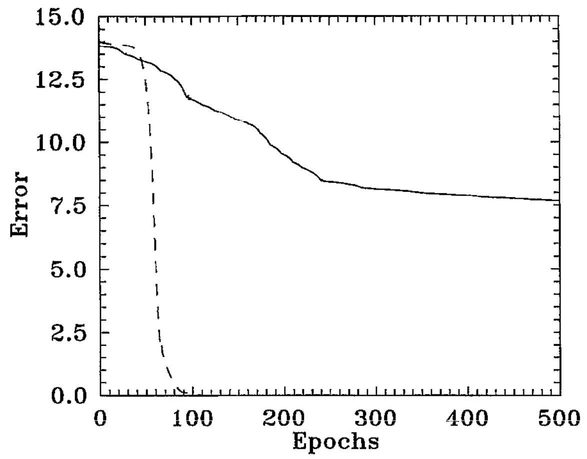
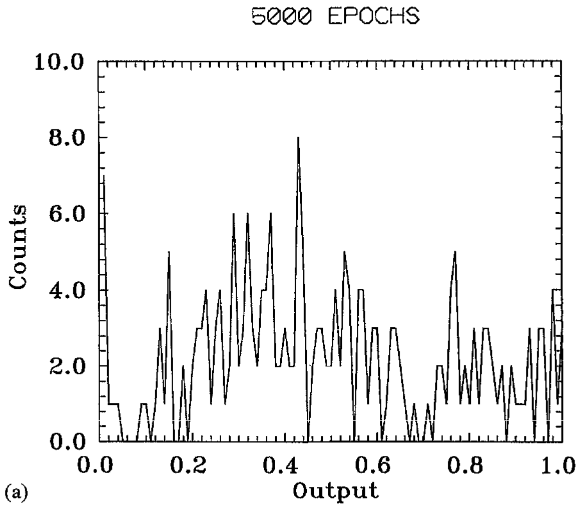


Fig. 7. The error function versus epoch numbers for two types of inputs. See solid curve for θ, ϕ and dashed curve for d_j.

(θ, ϕ) choice of inputs (see the solid curve) and the ANN does not converge even after 5 000 epochs (up to 500 epochs is shown in the figure). We have tried error minimization using various methods and also adjusted number of nodes in the hidden layer. However, the performance does not improve much. On the other hand, with the second choice of inputs as PCA transformed co-ordinates, the ANN converges very fast within 100 epochs. After the training is over, the weights are frozen and the method is applied to the test data. Fig. 8a and Fig. 8b show the ANN output distribution (after 5000 and 100 epochs, respectively, used during training) with two choices of inputs. Although the desired outputs are 0 and 1, the output values are broadly distributed between 0 and 1 when (θ, ϕ) are used as inputs. On the other hand, with d_j as inputs, the learning is not only faster, but also the output converges to either 0 or 1 after 100 epochs as shown in Fig. 8b. It is also noticed that even one node in the hidden layer is enough to achieve convergence. Therefore, we use a neural network with one node in the hidden layer with transformed co-ordinates d_j both for training and testing. We use different sets of test data having different number of true and false tracks. The idea is to test whether ANN can differentiate between true and false tracks which are inherently present after PCA test for global tracking as shown in Table 2 for $N = 100$. We apply the ANN to the

(a)


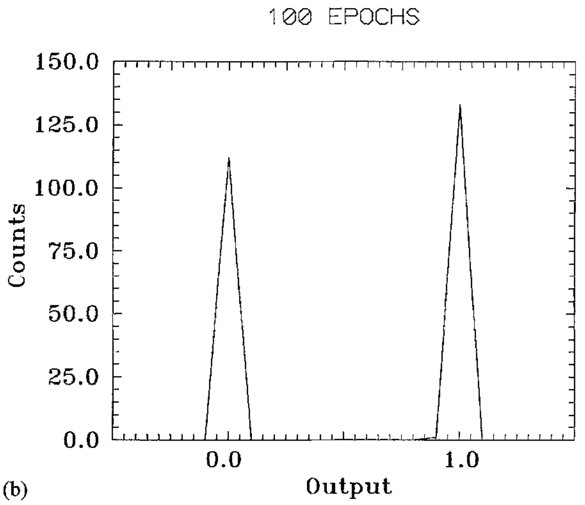
(b)


Fig. 8. (a) The ANN output distribution after 5000 epochs when the actual θ, ϕ track co-ordinates are fed as net inputs. (b) The ANN output distribution after 100 epochs when PCA transformed co-ordinates are fed as net inputs.

PCA output with different number of true and false tracks as well as with different multiplicities. The ANN results are shown in Figs. 9 and 10 along with PCA results for comparison. Fig. 9 shows the variation of true and false acceptance with multiplicity taking a typical $d_{\max}$ value of 9 and Fig. 10 shows the acceptances as a function of $d_{\max}$ for a given multiplicity 100 for both PCA and ANN. It can be seen that, although the ANN rejects some of

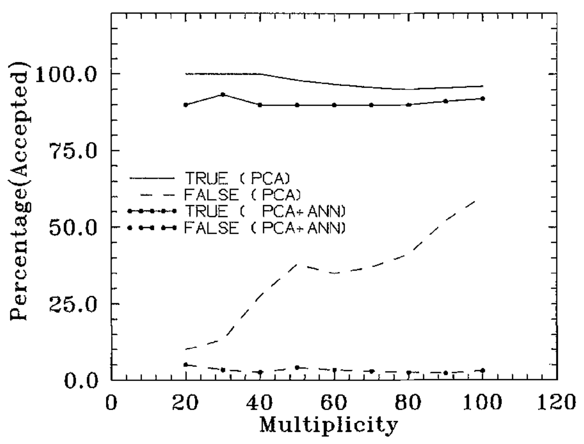


Fig. 9. The total number of true and false tracks as a function of track multiplicity both for PCA and ANN. The typical $d_{\max} \approx 10$.

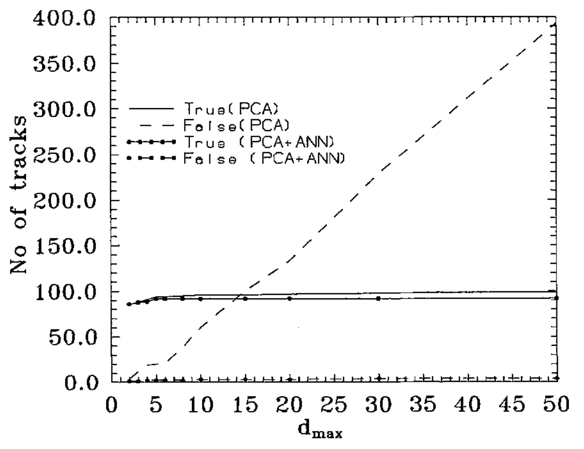


Fig. 10. The percentage of accepted true and false tracks at $N = 100$ as a function of $d_{\max}$ for global tracking both for PCA and ANN.

the true tracks, the false track contribution is drastically reduced to almost 1% or 2%.

The above neural network with one hidden node and one output node is quite simple as there is only one connection weight (W_{11}) from hidden to outer layer. This simple network is quite close to a two layer network in implementation as there is not much non-linearity from input to output except for the transfer functions. Although not shown explicitly, convergence can also be obtained even without any hidden layer. It may be mentioned here that the use of a two layer network is not new. Based on this approach a variety of applications have been discussed in the literature [9], which generally incorporate the idea of functional expansion of the inputs in a non-linear way. With this type of a simple percepton network, instead of a multi-layer percepton with many hidden nodes, the complexity of propagation of errors across the hidden layers is avoided and calculation of one point derivative at the output node is all that is required. With the PCA transformation (see Eq. (7)), the actual input layer of the network now acts as the hidden layer whose transformation weights are obtained from the principal component analysis. Therefore, the PCA transformation already implements one level of learning before the inputs are fed to the neural net. Further, the preprocessing in the present method is quite different and new as there is no further expansion on the input nodes. PCA does a simple linear co-ordinate transformation from one space to another. This transformation is also not arbitrary, rather the transformation matrix is obtained through PCA learning. Therefore, the above neural net with PCA transformed inputs incorporates two layers of learning; one at the PCA level and other one during the net learning.

7. Conclusions

The PCA and the ANN techniques have been applied for muon track recognition. It is found that the PCA method gives best performance when the track multiplicity is low. At higher multiplicities, the false track contribution increases which becomes quite significant if the number of tracking stations is reduced further. In the present analysis, whether it is 18 co-ordinates (with three layers per tracking station) or 12 co-ordinates (two layers per tracking station), the PCA performance does not differ much. The ANN method has also been applied for pattern recognition. It is found that the ANN cannot be trained with actual (θ, ϕ)

co-ordinates of the true and false tracks. However, the ANN learns much faster when the actual (θ, ϕ) co-ordinates are transformed to a new set of co-ordinates using PCA technique. This combined approach based on both PCA and ANN seems to give better performance. In fact, it is shown that with PCA transformation, even one node at the hidden layer is enough for the above application. Since less number of connection weights are involved, the complexity of the error propagation through the hidden layers can be avoided. In the above, the analysis has been carried out with the simulated data set generated from the PISA simulation code for the PHENIX muon tracking detectors. The above method is, however, quite general and can be applied for pattern recognition, particularly in situations when the track multiplicities are quite high and the PCA alone cannot eliminate all the false tracks.

Acknowledgements

We are thankful to Dr. S. S. Kapoor and Shri B. V. Dinesh for many useful discussions during the course of this work. We also express our gratitude to the PHENIX collaboration and the simulation group for the PISA simulation code used for this work.

References

[1] PHENIX Conceptual Design Report, 1993, The PHENIX Muon Arms: Current Design and Status, 1996, unpublished.
[2] PHENIX/SPIN Conceptual Design Report, 1995, unpublished.
[3] P. Chand, R.K. Choudhury, A.K. Mohanty, W. Davidson, C.F. Maguire, A. Rose, A. Trivedi, A primer for the PHENIX simulation code PISA and PISORP with an introduction to PADUA, Version 3.1, 23 February 1997, unpublished.
[4] M.G. Kendall, A. Stuart, The Advanced Theory of Statistics, 3rd ed., vol. 3, Ch. Griffin and Company, London, 1976, pp. 294.
[5] J. Hertz, A. Krogh, R.G. Palmer, Introduction to the Theory of Neural Computation, Lecture Notes, vol. I, Addison-Wesley, Reading MA, 1991.
[6] M. Riedmiller, H. Braun, Proc. IEEE Int. Conf. on Neural Networks (ICNN), San Fransisco, CA, 28 March – 1 April 1993, pp. 286–591.
[7] J. Reifman, J. Vitela, J.C. Lee, Topical Meeting on Nuclear Plant Instrumentation, Control and Man-Machine Interface Technologies, 18–21 April 1993, Oak Ridge, TE.
[8] C. Peterson, T. Rongvaldsson, L. Lonnblad, LU TP 93-29, CERN-TH 7135/94.
[9] Y.H. Pao, Adaptive Pattern Recognition and Neural Networks, Addison-Wesley, Reading MA, 1989.